UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/761,600	01/16/2001	Zakwan Shaar	190-1468	5823

23644 7590 10/22/2004

BARNES & THORNBURG
P.O. BOX 2786
CHICAGO, IL 60690-2786

EXAMINER	
HOGAN, MARY C	

ART UNIT	PAPER NUMBER
2123	

DATE MAILED: 10/22/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/761,600	SHAAR, ZAKWAN	
	Examiner	Art Unit	
	Mary C Hogan	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 10 September 2004.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 4-6,9-11,18,20 and 21 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 4-6,9-11,18,20 and 21 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 16 January 2001 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. **Claims 4-6. 9-11, 18, 20 and 21** have been examined and rejected.

Claim Objections

2. **Claims 18,20, and 21** are objected to because of the following informalities: the word "modeling" is misspelled. Appropriate correction is required.

Claim Interpretation

3. **Claim 4** recites "each message data structure includes a sender queue and receiver queue". Interpreting a message data structure to be directed to the format of the message being sent and the contents that are included, it is unclear how a message data structure can include a sender queue and a receiver queue. Therefore, this claim was interpreted to mean that the message data structure contains a *pointer* to a sender or a receiver process that have a queues in which the messages can be stored.

4. **Claims 4,10,11,and 18** are directed to scheduling a process on a scheduler queue and placing process type items on the queue. However, the specification and the prior art refer to processes as "specific functional units of the target system" **(specification, page 3)** that communicate to each other by way of messages. Therefore, it was unclear how a "process", being a representation of a functional unit of the target system, could be queued. From the specification's description of the message data structure **(specification, page 3)**, the attributes "sender[i]" and "receiver[I]" are described as pointers to a sender and receiver process, and combined with "data[I]" and "rcv_req_t[I]" form sender and receiver queues in which processes waiting to send or receive a message can be queued. From this description, the claims were interpreted to mean that scheduling a process and placing process type items on the queue refers to these pointers to sender and receiver processes that are included in the message data structure of a queued process.

5. **Claim 18** refers to "each item having a type value which indicated the item as being either a process-type item or a message-type item". In the specification, it is stated that the event type can be a message, process, activity or signal and that these events are handled in different ways according to their type. Therefore, it was concluded that the inclusion of this event type specifies how the communication between processes occurs between processes based on what event type is on the queue.

Claim Rejections - 35 USC § 102

6. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. **Claims 4, 5, 20 and 21** are rejected under 35 U.S.C. 102(b) as being anticipated by Lutter et al (Lutter et al, "Using VHDL for Simulation of SDL Specifications, IEEE, 1992), herein referred to as **Lutter**.

8. As to **Claim 4, Lutter** teaches: method according to Claim 20 wherein each message data structure includes a sender queue and a receiver queue, for queuing a plurality of sender and receiver processes **(page 630, column 2, section 2, sentence 5, page 632, column 2, lines 12-19)** wherein each message data structure (record) contains the process id of both the sender process and the receiver process, and each process has an input queue where arriving signals are stored.

5. As to **Claim 5, Lutter** teaches: a method according to Claim 20 further including checking the message data structure when a message has been processed, to determine whether there is at least one remaining sender and receiver in the message data structure for the message and, if so, rescheduling the message **(page 632, column 2, lines 41-50, page 633, column 1, lines 1 and 2)**. When the message from the SDL_behavior process is sent to the queue process, the queue process "processes" this signal to determine what signals can be sent back to the SDL_behavior process, encompassing "when a message has been processed". The queue process then checks the message data structures of signals in the queue to determine if a signal is acceptable to be sent to the SDL_behavior process, encompassing "checking the message data structure". Then, if an appropriate signal is found in the queue (such that the sender and receiver in the message data structure of a particular signal included the necessary information it needed to be an acceptable signal), the signal, or message, is sent to the SDL_behavior process. Since the message was already placed in the queue process, the message was already placed or scheduled on a queue. When the signal was found to be acceptable to be sent to the SDL_behavior process, it is sent to the input queue of the SDL_behavior process, therefore, rescheduled on a different queue.

9. As to **Claim 20, Lutter** teaches: a computer-implemented simulation method **(abstract)** comprising:

(a) modeling a target system as a set of processes that communicate with each other by way of messages **(page 630, column 2, paragraph2, sentence 3);**

(b) associating a message data structure with each of the messages **(page 632, column 2, items 1,2 and 3 and lines 12-19)** wherein the VHDL signal contains the process id of the receiver, the process id of the sender and contains a data slot for each type of signal;

(c) when a process requires to send a message, adding that process to the relevant message data structure as a sender process **(page 632, column 2, lines 12-22)** wherein when a signal is sent from one process to another, the process id of the sender process is included in the message data structure or record;

(d) when a process requires to receive a message, adding that process to the relevant message data structure as a receiver process **(page 632, column 2, lines 12-22)** wherein when a signal is sent from one process to another, the process id of the receiver process is included in the message data structure or record;

(e) scheduling a message for processing when there is at least one sender process and at least one receiver process in the message structure associated with the message **(page 632, column 2, lines 1-22 and lines 41-50)** wherein the sender process (process id of sender) and the receiver process (process id of receiver) that is contained in the VHDL signal are the sender and receiver processes in the data structure; and and the signals are sent to the queue from a process, therefore, scheduling the message for processing; and

(f) processing each scheduled message by calling the sender and receiver processes in the message data structure associated with the message **(page 632, column 2, lines 1-27)** wherein the sending of a signal (processing a scheduled message) calls the receiver of the signal using the process id of the receiver and calls the sender process by sending an acknowledge signal.

10. As to **Claim 21, Lutter** teaches: a data carrier, carrying a computer-readable program for performing a computer implemented simulation method comprising:

(a) modeling a target system as a set of processes that communicate with each other by way of messages**(page 630, column 2, paragraph2, sentence 3)**;

(b) associating a message data structure with each of the messages **(page 632, column 2, items 1,2 and 3 and lines 12-19)** wherein the VHDL signal contains the process id of the receiver, the process id of the sender and contains a data slot for each type of signal;

(c) when a process requires to send a message, adding that process to the relevant message data structure as a sender process **(page 632, column 2, lines 12-22)** wherein when a signal is sent

from one process to another, the process id of the sender process is included in the message data structure or record;

(d) when a process requires to receive a message, adding that process to the relevant message data structure as a receiver process **(page 632, column 2, lines 12-22)** wherein when a signal is sent from one process to another, the process id of the receiver process is included in the message data structure or record;

(e) scheduling a message for processing when there is at least one sender process and at least one receiver process in the message structure associated with the message**(page 632, column 2, lines 1-22 and lines 41-50)** wherein the sender process (process id of sender) and the receiver process (process id of receiver) that is contained in the VHDL signal are the sender and receiver processes in the data structure; and and the signals are sent to the queue from a process, therefore, scheduling the message for processing; and

(f) processing each scheduled message by calling the sender and receiver processes in the message data structure associated with the message **(page 632, column 2, lines 1-27)** wherein the sending of a signal (processing a scheduled message) calls the receiver of the signal using the process id of the receiver and calls the sender process by sending an acknowledge signal.

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. The factual inquiries set forth in *Graham* v. *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.

3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

13. **Claims 6 and 9** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Lutter** as applied to **Claim 20** above, and further in view of Hashmi (U.S. Patent 6,161,081), herein referred to as **Hashmi**.

14. As to **Claim 6, Lutter** teaches associating a message data structure with each of the messages **(page 632, column 2, items 1,2 and 3 and lines 12-19).**

15. **Lutter** does not expressly teach wherein each message data structure holds pointers to a composition activity, for composing a higher-level message from a lower level message, and to a decomposition activity, for decomposing a higher-level message into a lower-level message.

16. **Hashmi** teaches a message data structure holding pointers to a composition and decomposition activity **(column 2, lines 57-62, column 3, lines 5-6)**. Interface unit I1 is an interface unit that performs composition and decomposition of a message. **Hashmi** states that Unit U1 sends a message to the send port of the interface unit I1. The message data structure from U1, therefore, must contain a pointer to the send port of I1, and since I1 performs composition and decomposition, the message data structure sent by U1 contains a pointer to a structure that performs composition or decomposition activities. Further, **Hashmi** states that designs in VHDL contain functional units that communicate with each other by way of signals and that the functional units can be specified at different levels of complexity. Therefore, it is necessary to provide ways to allow these different levels of models to communicate with each other **(column 1, lines 5-40)**.

17. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the data structure as taught in **Lutter** to include a pointer to a composition and decomposition activity as taught in **Hashmi** since **Hashmi** teaches that designs in VHDL contain functional units that communicate with each other by way of signals and that the functional units can be specified at different levels of complexity, making it is necessary to provide ways to allow these different levels of models to communicate with each other **(column 1, lines 5-40)**.

18. As to **Claim 9, Hashmi** teaches: a method according to Claim 6 wherein the composition activity for a message is activated when a process is added to the message data structure as a receiver for that message **(column 3, lines 47-53)**. When the requesting core is ready to compose a message, it sends a request to the receiving core to send it the next lower-level data item for the composition. This sending of the message from the requesting core activates the composition activity. The request in the message to

have the next data item sent must include a pointer to the requesting core so that the receiving core knows where to send the data items. Therefore, a process (pointer to the requesting core) is added to the message data structure as a receiver since the requesting process will be receiving the lower level data for the composition activity. Further, **Hashmi** teaches: the decomposition activity for a message is activated when a process is added to the message data structure as a sender for that message **(column 3, lines 54-60)**. The decomposition activity for a message is activated when the decomposition request is added to the message data structure that is sent from the byte core to the nibble core. The data structure sent from the byte core to the nibble core also must include a pointer to the sender of the message, in this case, the byte core.

19. **Claims 10 and 11** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Lutter** as applied to **Claim 20** above, and further in view of Rayner (EP 0 854 429 A2), herein referred to as **Rayner**.

20. As to **Claim 10**, **Lutter** teaches each process has an input queue **(page 630, column 2, section 2, paragraph 1)** and teaches a simulation delta as a unit of time **(page 632, column 1, lines 20-25)**.

21. **Lutter** does not expressly teach that the queues are scheduling queues that are used for scheduling both messages and processes.

22. **Rayner** teaches the step of scheduling the messages for processing comprises providing at least one scheduler queue, which is used for scheduling both messages and processes **(column 3, lines 3-7)** and further teaches tow forms of simulated time: event time and delta time **(column 3, lines 8-13)**.

23. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the queues as taught in **Lutter** to be a scheduler queue as taught by **Rayner** since **Lutter** and **Rayner** both teach simulating a system in VHDL using processes that communicate with each other through messages (or signals) and since both teach delta time as a simulated time in the system as well as input queues for the processes. **Lutter** and **Rayner** are determined to be in the same field of art and describe similar simulation systems wherein **Rayner** explains a more detailed version.

24. As to **Claim 11**, **Rayner** teaches: a method according to **Claim 10** including the steps: scheduling the processes by placing process-type items on the scheduler queue and scheduling the messages by placing message-type items on the scheduler queue, processing each process-type item on the scheduler queue by calling the process to which the item relates and processing each message-type item on the scheduler queue by calling both the sender and receiver processes of the message to which the item relates **(column 11, lines 2-12).** From this description, an event is added onto the event queue. When the specified time for this event record to be processed arrives, the pop_events() function removes the

record from the event queue and calls the change_state() function of S1.Then, the pop_events() further calls the fire_event() function of S1 which in turn calls the enrty() function of the process P2. From this description, it is noted that when the event record is popped off the queue, both a message type event and a process type event resulted; therefore, both a process type item and a message type item were included in this event and scheduled on the queue.

25. **Claim 18** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Lutter**, in view of **Rayner** and further in view of Wilkes et al (Wilkes et al, "Application of High Level Interface-based Design to Telecommunications System Hardware", ACM, 1999), herein referred to as **Wilkes**.

26. As to **Claim 18**, **Lutter** teaches: a computer-implemented simulation method comprising the steps: modeling a target system as a set of processes that communicate with each other by way of messages **(page 630, column 2, paragraph2, sentence 3)**; associating the messages with sender and receiver processes **(page 632, column 2, items 1,2 and 3 and lines 12-19)** wherein the VHDL signal contains the process id of the receiver, the process id of the sender. Further, **Lutter** teaches each process has an input queue **(page 630, column 2, section 2, paragraph 1)** and teaches a simulation delta as a unit of time **(page 632, column 1, lines 20-25)**.

27. **Lutter** does not expressly teach that the queues are scheduling queues that are used for scheduling both messages and processes and that the items have a type value which indicates the item as being either a process type item or a message type item.

28. **Rayner** teaches providing at least one scheduler queue, holding a series of items (**column 3, lines 3-7**). **Rayner** further teaches: scheduling the processes by placing process-type items on the scheduler queue and scheduling the messages by placing message-type items on the scheduler queue, processing each process-type item on the scheduler queue by calling the process to which the item relates and processing each message-type item on the scheduler queue by calling both the sender and receiver processes of the message to which the item relates (**column 11, lines 2-12).** From this description, an event is added onto the event queue. When the specified time for this event record to be processed arrives, the pop_events() function removes the record from the event queue and calls the change_state() function of S1.Then, the pop_events() further calls the fire_event() function of S1 which in turn calls the enrty() function of the process P2. From this description, it is noted that when the event record is popped off the queue, both a message type event and a process type event resulted; therefore, both a process type event and a message type event were included in this event and scheduled on the queue.

29. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the queues as taught in **Lutter** to be a scheduler queue as taught by **Rayner** since **Lutter** and

Rayner both teach simulating a system in VHDL using processes that communicate with each other through messages (or signals) and since both teach delta time as a simulated time in the system as well as input queues for the processes. **Lutter** and **Rayner** are determined to be in the same field of art and describe similar simulation systems wherein **Rayner** explains a more detailed version.

30. **Wilkes** teaches an interface in a simulation system and describes the interface as encapsulating data types with how they are communicated **(page 2, column 1, paragraph 1)**. When a message is sent through the interface, it includes a parameter specifying the data type **(section 3.1, first two paragraphs)**. From this explanation, the message that was queued to be sent through the interface contained a parameter, "type" and this governs how the message is communicated between processes. It is concluded that this explanation is concurrent with the claim interpretation as discussed above **(see paragraph 5)** in that the "type" parameter in the message data structure determines how the event will be handled. Further, **Wilkes** teaches that this interface is implemented in VHDL+ as an improvement to the VHDL language to improve the design process where difficulties were encountered when interfaces between the blocks in the design were at different abstraction levels. The interfaces, or translation tools, written in VHDL took too long to write and check and the interfaces in VHDL+ solved this problem by reducing time to market **(Introduction and Section 8.1, paragraph 1)**.

31. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the message data structure as taught in **Lutter** to include an item type as discussed in **Wilkes** since the item type encompasses how these data types are communicated as taught by **Wilkes (page 2, column 1, paragraph 1)**. Further, since **Lutter** is directed to a simulation implemented in VHDL, and **Wilkes** is directed to an improvement to VHDL concepts, it would have been obvious to modify the invention as taught in **Lutter** to include these improved concepts as taught in **Wilkes** to reduce the time to market.

Response to Arguments

32. Applicant arguments filed on 9/10/04 regarding Claims 4-6, 9-11, 18, 20 and 21 have been considered, but they are not persuasive.

33. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.,* 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

34. Applicant disagrees that **Lutter** teaches "…when a process requires to send a message, the process is assigned to the relevant message data structure as a sender, and when a process is required to receive a message, the process is assigned to the relevant message data structure as a receiver" and further, applicant states "…rather than adding processes to message data structures, **Lutter** adds signals to process data structures, which is a completely different thing" **(page 6, paragraph 3)**.

35. As to the arguments stated above, **Lutter** discusses the make up of the message data structure sent between processes. These data structures include the process id of the receiver and the process id of the sender. The presence of the process id of the sender and the receiver in the record or data structure is part of the typical construct of signals in VHDL **(page 632, column 2, lines 1-27)**. The signal states which process is the receiver of the message (or requiring to receive a message), therefore assigning this process to the data structure as the process id of the receiver, and the signal states which process is the sender of the message (or requiring to send a message), therefore, assigning this process to the message data structure as the process id of the sender. Further, the record, or data structure, includes a data slot for each *type* of signal as well as the process id's of the receiver and sender processes. The message data structure is determined to be the *makeup* or *structure* of a VHDL signal, therefore, not a signal itself. **Lutter** adds the signal *types* to the message data structure, not signals.

36. Applicant states, "…the only queues described by **Lutter** are the input queues in which the arriving signals are stores for processes. Thus, **Lutter** queues signals, not processes, and hence clearly does not queue sender and receiver processes", **(page 7, first paragraph)**.

37. As to the argument stated above, **Lutter** teaches that each message data structure (record) contains the process id of both the sender process and the receiver process **(page 632, column 2, lines 1-27)**. **Lutter** further teaches that processes communicate with each other through signals (messages) and that and each process has an input queue where arriving signals are stored **(page 630, section 2, paragraph 1)**. As discussed above **(paragraph 4)**, it is unclear how a process can be queued. However, the signals or messages that are queued contain pointers to processes since this is necessary for the signals to be sent from one process to another, therefore, the presence of the process id's in the queued signals explains how pointers are "queued" as interpreted from the claim language.

38. Applicant states "…**Lutter** does not describe a message data structure containing queues for sender and receiver processes: the only queue described by **Lutter** is for signals, not processes" **(page 7, paragraph 4)**.

39. As to the argument stated above, **Lutter** describes the message data structure containing pointers to queues for sender and receiver processes **(page 632, column 2, lines 12-19)** wherein the process id of

the sender and receiver are pointers to the input queues of the processes that the message is sent from and sent to **(see also, paragraph 3 above)**. The queues of the processes containing these signals, or message data structures, therefore include signals and processes (in the form of pointers).

40. Applicant states, "**Hashmi** clearly does not teach a message data structure holding pointers to a composition activity and a decomposition activity" **(page 7, paragraph 7)**.

41. As to the argument stated above, **Hashmi** teaches a message data structure holding pointers to a composition and decomposition activity **(column 2, lines 57-62, column 3, lines 5-6)**. Interface unit I1 is an interface unit that performs composition and decomposition of a message. **Hashmi** states that Unit U1 sends a message to the send port of the interface unit I1. The message data structure from U1, therefore, must contain a pointer to the send port of I1, and since I1 performs composition and decomposition activities, the message data structure sent by U1 contains a pointer to a composition or decomposition activity.

42. Applicant states, "**Hashmi** clearly does not teach that the decomposition activity is activated when a process is added as a sender for the message and the composition activity is activated when a process is added as a receiver" **(page 7, paragraph 8)**.

43. As to the argument stated above, **Hashmi** teaches the composition activity for a message is activated when a process is added to the message data structure as a receiver for that message **(column 3, lines 47-53)**. When the requesting core is ready to compose a message, it sends a request to the receiving core to send it the next lower-level data item for the composition. This sending of the message from the requesting core activates the composition activity. The request in the message to have the next data item sent must include a pointer to the requesting core so that the receiving core knows where to send the data items. Therefore, a process (pointer to the requesting core) is added to the message data structure as a receiver since the requesting process will be receiving the lower level data for the composition activity. Further, **Hashmi** teaches: the decomposition activity for a message is activated when a process is added to the message data structure as a sender for that message **(column 3, lines 54-60)**. The decomposition activity for a message is activated when the decomposition request is added to the message data structure that is sent from the byte core to the nibble core. The data structure sent from the byte core to the nibble core also must include a pointer to the sender of the message, in this case, the byte core.

44. Applicant states, "In **Rayner**, it is clear that the event queue and the delta queue are used only to schedule changes of state of signal objects" and that "**Rayner** does not teach scheduling messages by calling the sender and receiver processes" (page 8, paragraphs 4,5 and 7).

45. As to the argument above, **Rayner** teaches that the event and delta queues are used to schedule changes to the state of the model **(column 3, lines 3-4)**. Since the simulation model includes both signals and processes that are changeable parts of the model **(Figure 2 and description)**, the changes that are scheduled on the queue are used to schedule changes to the processes and signals (or messages). Further, **Rayner** teaches scheduling the processes by placing process-type items on the scheduler queue and scheduling the messages by placing message-type items on the scheduler queue, processing each process-type item on the scheduler queue by calling the process to which the item relates and processing each message-type item on the scheduler queue by calling both the sender and receiver processes of the message to which the item relates **(column 11, lines 2-12).** From this description, an event is added onto the event queue. When the specified time for this event record to be processed arrives, the pop_events() function removes the record from the event queue and calls the change_state() function of S1.Then, the pop_events() further calls the fire_event() function of S1 which in turn calls the enrty() function of the process P2. From this description, it is noted that when the event record is popped off the queue, both a message type event and a process type event resulted; therefore, both a process type event and a message type event were included in this event and scheduled on the queue.

46. Applicant states in reference to **Wilkes**, "there is clearly no suggestion in this that q_cmd_t indicates the type of an item in a queue, and in particular, whether that item is a process-type item or a message type item" **(page 9, paragraph 2)**.

47. As to the argument above, **Wilkes (sections 3 and 3.1)** teaches that an interface is a design unit that sends messages between two processes. MESSAGE is the message that is sent from one process to another (between client and server, in this example) through the interface. Since this message was sent from a sender process (client), it was therefore contained in the queue of the sender process (client) before being sent to the interface. In the data structure of this message is a parameter of type q_cmd_t. **Wilkes** teaches that this interface encapsulates data types with how they are communicated. Since the "type" referred to in Claim 18 was determined as directed to how the events are handled in different ways according to their type (see paragraph 5), it was concluded that the inclusion of this event type in MESSAGE specifies how the communication between processes occurs for this "type", just as is specified in the claim. Therefore, if the type was specific to a message type item or a process type item, the interface would communicate the message accordingly since the interface encapsulates the data type and how it is communicated.

Conclusion

48. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

49. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

50. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary C Hogan whose telephone number is 703-305-7838 until 10/28/04 or 571-272-3712 after 10/28/04. The examiner can normally be reached on Monday-Friday 7:30AM-5PM. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kevin Teska can be reached on 703-305-9704. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary C Hogan
Examiner
Art Unit 2123

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER

***